CLAIMS

We claim:

1. An architecture for a pipeline processor, comprising :

a plurality of stages ;

a forwarding network of forwarding paths which connect said stages; and

a register file for operand write-back, wherein one of said stages includes an optimization-of-power-consumption function via inhibition of writing and subsequent readings in said register file of operands retrievable from said forwarding network because of reduced liveness lengths of said operands.

2. An architecture according to claim 1 wherein said function is configured for performing selectively, for a register of said register file assigned by a first instruction comprising a write-back stage and used by a second instruction, the following:

disabling of write-back of said register in said register file in the write-back stage of said first instruction; and

inhibiting assertion of a read address of said register in said register file by said second instruction.

3. An architecture according to claim 1 wherein said one of said stages includes dedicated logic for disabling the write-enable signals that enable writing in said register file.

4. An architecture according to claim 3 wherein the one of said stages is a decoding stage for decoding the instructions and reading the operands from said register file, and said dedicated logic is included in said decoding stage.

5. An architecture according to claim 1, further comprising dedicated logic which minimizes read-port switching activity in said register file by maintaining values on the input read addresses of the register file at previous clock cycles.

6. An architecture according to claim 1 wherein said processor is a superscalar processor comprising a hardware control unit capable of analyzing an instruction window to determine liveness lengths of registers of the register file.

7. An architecture according to claim 1 wherein said architecture is configured as a VLIW architecture, in which a decision of activating said function is delegated to a compiler.

8. An architecture according to claim 7 wherein the compiler transfers information to hardware control logic, reserving specific operation bits in instruction encoding.

9. An architecture according to claim 7 wherein the compiler transfers information to hardware control logic, exploiting unused instruction encoding bits.

10. An architecture according to claim 1, further comprising interstage registers comprised between the stages for storing a volatile architectural state, and wherein the architecture is configured for discarding elements that exit said volatile architectural state, avoiding write-back in said register file.

11. An architecture according to claim 10 wherein the architecture is adapted to operate on instructions configurable as exceptions, and , in order to ensure re-execution of instructions constituting an exception in a correct processor state, write-back is envisaged of values inhibited as regards writing in said register file in the presence of a signal that is configured as an exception.

12. An architecture according to claim 11 wherein the plurality of stages includes it comprises a decoding stage for decoding instructions and reading operands from said register file, an instruction-execution stage, and a memory-access stage, and said write-back is envisaged whenever an exception signal is generated in one of said stages.

13. An architecture according to claim 11 wherein, in the presence of an instruction configured as an exception, the architecture is configured for executing the instructions in the pipeline until their completion, there being envisaged write-back, in said register file, of the results of all the instructions in the pipeline.

14. An architecture according to claim 1, further comprising interstage registers coupled between stages of the plurality of stages, including latch registers used as a memory layer for storing the operands.

15. An architecture according to claim 14 wherein said interstage registers are configured in such a way that they are visible to the compiler and are not visible to the programmer.

16. An architecture according to claim 14 wherein said interstage registers are not write-addressable, in so far as they are implicitly addressed.

17. An architecture according to claim 14 wherein said interstage registers are configured as a transient memory which cannot be associated to a machine state that can be saved in the event of an exception.

18. An architecture according to claim 17 wherein said architecture is configured in such a way that sequences of instructions that use said interstage registers are treated as atomic sequences that are not subject to interrupts.

19. An architecture according to claim 18 wherein disabling of any interrupt is envisaged prior to start of said sequences, and a machine state is rendered stable prior to interrupt re-enabling by means of write-back in the register file or in the memory.

20. An architecture according to claim 17, further comprising a function of generation of two pseudo-instructions, one for checkpoint declaration and one for checkpoint release, with the provision of a shadow register, wherein a program counter is saved from an instant of checkpoint declaration, a machine state not being modifiable until checkpoint release, whereby, upon checkpoint release, the shadow register is reset and the interrupts are disabled atomically.

21. An architecture according to claim 20 wherein results computed between said two pseudo-instructions are entrusted to a real state of the processor with subsequent interrupt re-enabling to enable re-start of normal execution.

22. An architecture according to claim 20 wherein, in the presence of interrupts between said pseudo-instructions, the execution is made to restart, after handling of the interrupts, starting from the program counter stored in the shadow register.

23. An architecture according to claim 20 wherein all register writings comprised between said pseudo-instructions involve only said interstage registers, whereby said register file is involved only for data reading.

24. An architecture according to claim 20 wherein the register file includes a subset reserved for transient variables that are generated between said two pseudo-instructions and a liveness length of which exceeds a maximum value allowed by the pipeline.

25. An architecture according to claim 24 wherein the first appearance of transient registers in a sequence being checkpointed is a definition such as a load or write in a register, which can be seen as a constituent part of the machine state after checkpoint release.

26. A method of reducing power consumption in a processor architecture that includes a plurality of stages, a forwarding path that connects the stages, and a register file for operand write-back, the method comprising:

determining a liveness length of an operand in the forwarding path; and

inhibit writing of the operand to the register file if the liveness length of the operand is shorter than a predetermined value.